Journal of Parallel and Distributed Computing–1459

JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 52, 40–68 (1998)
ARTICLE NO. PC981459

Performance Improvement of Allocation Schemes for Mesh-Connected Computers[1]

Chung-yen Chang and Prasant Mohapatra

Department of Electrical and Computer Engineering, Iowa State University, Ames, Iowa 50011
E-mail: chungyen@iastate.edu. prasant@iastate.edu

Received May 12, 1997; accepted April 3, 1998

Fragmentation is the one of the main problem in scheduling jobs on multicomputer systems. In this paper, we classify the fragmentation problems in multicomputer systems on the basis of their causes. It is noted that among all kinds of fragmentation, physical fragmentation is seldom addressed by contemporary allocation schemes. Two novel processor allocation schemes that reduce physical fragmentation for the mesh-connected multicomputers are proposed to improve the performance of the mesh-connected computers. The first scheme allows jobs to be executed on a reduced size submesh adaptively. The second scheme allocates jobs on noncontiguous processors if fragmentation prevents a job from being allocated. Extensive simulations indicate significant performance gain by using these two schemes. The performance improvement along with flexibility of implementation make these two allocation schemes suitable for mesh-connected multicomputers. © 1998 Academic Press

Key Words: adaptive noncontiguous allocation; mesh-connected multicomputers; multicomputer performance; processor allocation; restricted size reduction.

1. INTRODUCTION

Multicomputer systems are cost-effective alternatives of the traditional supercomputers. Among the topologies proposed for multicomputer architectures, the mesh

[1] This research was supported in part by the National Science Foundation under Grants MIP-9628801, CCR 9634547, and CDA 9617375. A preliminary version of the RSR scheme was presented at the International Conference on Distributed Computing Systems, 1996.

0743-7315/98 $25.00

topology has gained popularity because of its simplicity, regularity, and ease of VLSI implementation. It has been widely adopted in several commercial and experimental systems. Processors in a multicomputer are shared among different processes. Jobs are allocated to a subset of processors for execution. In a practical environment, multicomputers support a diverse mix of applications of various sizes and characteristics in a dynamic fashion. To support such an environment, it is essential to implement an efficient processor allocation scheme.

Conventional allocation schemes for mesh-connected multicomputers [1–6] attempt to locate a contiguous portion of the processors for the execution of a job to minimize the distance of interprocessor communication path and to avoid the interprocess interference. The processors allocated to a job retain the same topology as the entire system with the number of processors determined according to the requirement of individual jobs. In a mesh-connected system, jobs are allocated to submeshes. A job retains all the nodes of the submesh for the entire duration of its execution. Once a job is allocated, it runs till completion. It is the processor allocator's responsibility to detect free nodes and to decide whether the free nodes can form a submesh for the execution of a job. Allocation algorithms with better recognition ability for available submeshes can improve the chance of assigning a job into the system and reduce the waiting delay. However, as studies showed [8, 9], a significant performance improvement cannot be obtained by refining the conventional allocation algorithms. A multicomputer is typically underutilized when operated in a dynamic environment because of the fragmentation problems. Fragmentation occurs when there are free nodes in the system but the allocation algorithm fails to allocate these nodes to the waiting jobs. This is the main limiting factor of the performance of contemporary allocation schemes. In order to improve the performance of multicomputer systems, fragmentation has to be minimized.

We evaluate two novel processor allocation schemes which attempt to alleviate the fragmentation problems. Both schemes try to allocate a job to a submesh of the same size if available. When the required size submesh is not available, the first method allocates a job to a smaller size submesh if it is available and therefore avoids fragmentation. This scheme is called *restricted size reduction* (*RSR*) allocation [10]. The tradeoff is the increased execution time of reduced size jobs due to the fact that the number of processors allocated to these jobs are less than the required number. The second scheme adaptively splits jobs into small subframes for allocation. The subframes of a job are allocated simultaneously to noncontiguous locations. Because small subframes are easy to allocate, the probability of allocating a job is increased. This scheme is therefore called *adaptive noncontiguous allocation* (*ANCA*). The tradeoff for the ANCA scheme is the increased communication latency caused by the noncontiguous allocation.

We compare the performance of these two schemes along with other allocation schemes proposed earlier for the mesh-connected multicomputers. Extensive simulations are conducted to measure the system fragmentation and the average turnaround time of a job. The tradeoffs for both schemes are also studied. The simulation results show a promising performance improvement that is achievable with the RSR and ANCA schemes.

Related works in processor allocation are surveyed in the next section followed by a classification of fragmentation in Section 3. The motivation factors behind the two schemes are outlined in Section 4. Detailed descriptions of the RSR and ANCA schemes are presented in Section 5 and 6, respectively. Performance evaluation of the two schemes is shown in Section 7.

2. PREVIOUS ALLOCATION SCHEMES

Processor allocation schemes have been extensively studied by several researchers. Conventional allocation schemes focus on finding the available submesh according to the request of a job. A few recent approaches eliminate the constraint of allocating jobs to the requested size submesh. We outline several allocation schemes proposed in the literature.

2.1. Contiguous Allocation Schemes

Two Dimensional Buddy: The two dimensional buddy (*TDB*) scheme [1] is a generalization of the one dimensional buddy algorithm used for storage allocation. The system is assumed to be a square with side lengths equal to a power of two. The size of a requested submesh is rounded up to a square with sidelengths as the nearest power of two while being allocated to the system. A square submesh can form a larger square submesh with its three neighboring buddies. Jobs are allocated to buddies of submeshes. A bit-array is used to keep track of the allocation and deallocation processes. The TDB allocation suffers from internal fragmentation because it only allocates square submeshes whose side lengths are equal to a power of two. It is also not applicable for commercial systems such as the Intel Touchstone Delta and the Intel Paragon because of its square shape requirement.

Frame Sliding: The frame sliding (*FS*) method [2] is proposed to reduce the fragmentation problem of the TDB allocation. It can be used for meshes of any arbitrary size. The requested submesh of a job is called a frame. The algorithm slides the frame across the system to examine for a free submesh to execute the job. Two bit-arrays are used, one to keep track of the allocated nodes and the other to scan for the available submesh.

First-fit and Best-fit: The first-fit and best-fit algorithms proposed in [3] guarantee the recognition of asubmesh, provided it exists. It scans the entire mesh for possible allocation. Bit arrays are used for scanning of available processors.

Adaptive-Scan: To further enhance the allocation ability, the adaptive-scan [4] changes the orientation of a submesh when the required submesh in the original orientation is not available. By rotating the orientation of the submesh request, the possibility of allocating a job is increased and hence the performance is improved.

Busy-List: The busy-list allocation scheme [5] is different from the other allocation algorithms in the sense that it uses a busy-list for checking the availability of processors instead of the bit-array used by other algorithms. It has the same submesh recognition ability as that of the adaptive-scan policy.

Free-List: Another algorithm using linked-list for allocation is proposed in [6]. The free-list scheme maintains a list of free submeshes for the allocation. Its allocation process has a lower time complexity than that of the busy-list scheme but its deallocation time complexity is higher. Its submesh recognition ability is similar to that of the adaptive-scan and busy-list algorithms.

Limit: Limit allocation [9] was proposed for the hypercube system and later extended to the mesh topology in two independent works [10, 12]. The idea is to allocate a job to a smaller size submesh if the required submesh cannot be allocated. Probability of finding a smaller size submesh is larger and therefore the average turnaround time of jobs is reduced.

The above allocation schemes consider only contiguous regions for the execution of a job and are referred to as contiguous allocations. Distance of the communication path is expected to be minimized in contiguous allocations. Only messages generated by the same process are expected within a submesh and therefore cause no intertask contention in the interconnection network. On the other hand, the restriction that jobs have to be allocated to contiguous processors reduces the chance of successfully allocating a job. It is possible for the system to fail to allocate a job while a sufficient number of processors are available.

2.2. *Noncontinuous Allocation Schemes*

Hardware advances such as wormhole routing and faster switching techniques have made the communication latency less sensitive to the distance between the communicating nodes. This makes allocating a job to noncontiguous processors plausible. Allocation of jobs to noncontiguous nodes allows jobs to be executed without waiting if the number of available processors is sufficient. Several noncontiguous allocation algorithms are proposed in [7] including *random*, *naive*, and *Multiple Buddy System* (MBS).

The random allocation is a straightforward strategy in which a job requesting p processors is satisfied with p randomly selected nodes. The naive algorithm allocates a request for p processors to the first p free nodes found in a row major scan. The multiple buddy strategy is an extension of the 2-D buddy strategy. The system is divided into non-overlapped square submeshes with side lengths equal to powers of two upon initialization. The number of processors, p, requested by an incoming job isfactorized into a base four representation of $\sum_{i=0}^{\lfloor \log_4 p \rfloor} d_i \times (2^i \times 2^i)$, where $0 \leqslant d_i \leqslant 3$. The request is then allocated to the system according to the factorized number in which d_i number of $2^i \times 2^i$ blocks are required. If a required block is unavailable, MBS recursively searches for a bigger block and repeatedly breaks it down into buddies until it produces blocks of the desired size. If that fails, the requested block is broken into four requests for smaller blocks and the searching process repeats. The random and naive allocations ignore the contiguity of nodes allocated to a job and may incur substantial increases in the communication delay.

It is observed in [7, 15] that partially noncontiguous allocations that preserve some degree of contiguity provide better performance than the totally noncontiguous allocations. This is because of the effect of message contention. Despite the fact that the communication latency is expected to be less sensitive to the distance of the communication path with today's technology, the contention for the communication links among different messages introduces extra delay that cannot be ignored. Geometry of allocated nodes (relative positions of the nodes) is another factor for message contention. Parallel algorithms and applications may be optimized for the communication pattern in the architecture used. As indicated in [14], a proper mapping of processes onto processors is critical to the communication latency. The violation of submesh geometry introduces communication overhead due to the change in the optimized communication pattern. Therefore, the geometry should be retained among the allocated processors as much as possible. None of the past noncontiguous allocation policies have addressed this issue. The possibility of saturating the interconnection network due to the message contention makes the previously proposed noncontiguous allocation schemes less attractive. The ANCA scheme is proposed to take advantage of noncontiguous allocation while avoiding the above problems.

3. FRAGMENTATION PROBLEM

Fragmentation prevents the idle processors from being utilized. It can be classified as internal and external fragmentation. The *internal fragmentation* is a result of allocating jobs only to certain size submeshes. When a job is assigned to more processors than it requires, the extra nodes allocated are not used for actual computation and are wasted. This happens when a job requests a submesh that does not fit the requirement of the allocation algorithm. As an example, internal fragmentation occurs when a job does not require a square submesh with sides equal to a power of two and is allocated using the TDB scheme.

External fragmentation occurs when the allocation scheme cannot allocate the available processors to the incoming jobs. It can be further divided into three types based on their causes. The first type of external fragmentation is called *insufficient resource fragmentation.* Every job requires a certain number of processors for its execution. If the available processors in the system are less than the number required, the job cannot be allocated. The second type of external fragmentation is a result of imperfect recognition ability of the allocation algorithms. A suitable submesh may exist for the execution of a job while the allocation algorithm fails to locate the available submesh. This type of external fragmentation can be found in the TDB and the frame-sliding algorithms (if the frame slides through more than one hop at a time). We refer to this type of external fragmentation as *virtual fragmentation.* The third type of external fragmentation is caused by dynamic departures of jobs. Nodes released by the terminated jobs can be scattered in the mesh. They may not form a submesh big enough to accommodate the incoming

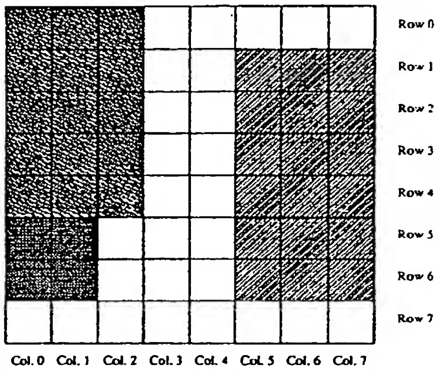

FIG. 1. Fragmentation problems.

task although the number of free nodes may be sufficient. This type of fragmentation is therefore referred to as *physical fragmentation*. Figure 1 shows a mesh system with three busy submeshes highlighted in different shades. There are 27 free nodes in this mesh. Using the conventional schemes, any job that requires more than 27 nodes cannot be allocated because of insufficient resource fragmentation. Suppose a job requires a 3 × 2 submesh (3 columns, 2 rows). Using the frame-sliding algorithm (where the frame slides by the height and width of the required submesh size), it will miss the free 3 × 2 submesh in the mesh because of its imperfect recognition ability. This is an example of virtual fragmentation. An example of physical fragmentation can be observed from the same figure when a 4 × 5 submesh is required. The 27 available nodes do not form the required shape and therefore will not be allocated with conventional allocation schemes.

TABLE 1

Fragmentation of Various Allocation Schemes

Scheme	Internal fragmentation	Insufficient resource	Virtual fragmentation	Physical fragmentation
TDB	Yes	Yes	Yes	Yes
Frame-sliding	No	Yes	Yes[a]	Yes
FF and BF	No	Yes	Some[b]	Yes
Adaptive-scan	No	Yes	No	Yes
Busy-list	No	Yes	No	Yes
Free-list	No	Yes	No	Yes
RSR	No	Some[c]	No	Some[c]
Noncontiguous	No	Yes	No	No
ANCA	No	Yes	No	Some[d]

[a] If the window skips through nodes while sliding.

[b] Only happens when the available submesh is in a different orientation.

[c] Can be completely removed if size reduction allows all jobs to be executed on a single node.

[d] Can be completely removed with adaptability set to allow all jobs to be allocated non-contiguously.

The above allocation schemes and their associated fragmentation are listed in Table 1. The performance of an allocation scheme is inversely proportional to the fragmentation it creates in the system. TDB is the only scheme listed with all fragmentation. Its performance is also the worst compared to the other allocation schemes in a dynamic system environment. Frame-sliding has physical and may have virtual fragmentation. The first-fit and best-fit algorithms have virtual fragmentation only when the submesh request and the available submesh have different orientations. Their performance is slightly worse than the other allocations that do not have virtual fragmentation. However, all conventional algorithms cannot deal with physical fragmentation. RSR, noncontiguous allocation, and ANCA are the only schemes that can reduce the physical fragmentation.

4. MOTIVATING FACTORS

The turnaround time of a parallel application can be represented as a sum of three components,

$$T_{\text{total}} = T_{\text{queue}} + T_{\text{comp}} + T_{\text{comm}}. \qquad (1)$$

The queuing delay, T_{queue}, represents the time a job spends in the queue. It is a function of the allocation algorithm and the scheduling strategy. The execution time of a job consists of computation time, T_{comp}, and communication latency, T_{comm}. The computation time depends on the amount of computation required for an application. It also depends on the amount of parallelism and the number of processors allocated to a job. The communication latency is the time a job spent waiting for different subtasks to exchange information. It is a function of the communication pattern of application and the positions of allocated nodes.

To improve the performance of a multicomputer system, the three delay components have to be minimized. Contiguous allocations focus on minimizing T_{comm}. Because of physical fragmentation associated with contiguous allocation schemes, the system is underutilized and a large T_{queue} is observed. The queuing delay becomes dominant in the turnaround time of a job when the system load gets higher. The average turnaround time of jobs can be reduced if T_{queue} can be limited. The RSR scheme allocates jobs to a smaller size submesh when the submesh of the required size cannot be located. The queuing delay is therefore reduced at the penalty of longer execution time. The ANCA scheme utilizes physically fragmented nodes in a mesh to avoid unnecessary waiting and therefore introduces overhead into the communication latency. If the increase in execution time or communication latency is less significant than the decrease in queuing delay, the total turnaround time of a job can be reduced.

5. RESTRICTED SIZE REDUCTION (RSR) SCHEME

To reduce the effect of fragmentation, the RSR scheme adaptively allocates jobs to smaller size submeshes when fragmentation prevents them from being allocated.

The tradeoff for this scheme is between the queuing delay and the longer execution time caused by executing jobs on a smaller number of processors. Size reduction is a straightforward process in which a submesh is folded along its longer side. Executing a job on a smaller submesh adaptively benefits the performance in two ways. First, physically fragmented nodes are utilized which, in turn, helps in accommodating a greater number of jobs in the system. Physical and insufficient resource fragmentations are thus reduced. Second, the waiting delay is reduced because jobs can be allocated earlier. Quite often it might be advantageous to execute a task paying a penalty of execution time than to wait for the availability of the required size and shape of the submesh. The RSR scheme exploits this observation and has the following important characteristics.

1. The RSR scheme is a generic processor management concept and is not limited to a single architecture or a particular allocation algorithm.

2. It is fair because jobs are serviced in a first-come-first-serve order.

3. It is adaptive. A job is only folded when fragmentation prevents it from execution. A job is always assigned the system resources it requested when the resources are available. This property guarantees that the system maintains a reasonable utilization at all ranges of workload.

4. It is flexible. System administrators can determine the optimal restriction on the size reduction according to the individual system's need and workload. An individual job can also specify the number of size reductions it can tolerate to avoid performance degradation.

5. It has low storage and computation complexity compared to the other approaches.

5.1. Suitability of Size Reduction

The suitability of size reduction needs to be addressed for the practical usage of the RSR scheme. In a multicomputer system, jobs come in different sizes according to their inherent parallelism and resources requests. The inherent parallelism of a job prohibits changing the job size randomly. However, we argue that it is safe to scale down a job in a regular fashion. The degree of parallelism limits the maximum number of subtasks that can be run in parallel. It is always more possible to reduce the number of concurrently running subtasks than to increase it. The nCUBE's software environment [16] explicitly supports execution of a job on different sized cubes. A program can be executed on a subcube larger or smaller than the one specified when it was compiled. For applications that require at least some certain number of processors to execute, it is still possible to fold the program and run them on the smaller subsystem in a context-switching fashion. However, the amount of size reduction is restricted by the memory requirement of the job and the memories available at the nodes. The RSR algorithms never reduce the size requirement of a job if the available memory is insufficient.

Executing jobs on a reduced size submesh causes execution time to be longer. General speedup studies [17, 18] state that for most of the applications, parallel computers provide sublinear speedups. Therefore, when a job is folded to half of the size it requested, its execution is likely to be less than doubled. Additionally, the communication overhead for less processors is expected to be reduced for several reasons. First, the communication path is shorter in a smaller subsystem. Second, the smaller number of processors in the smaller subsystem causes less interference between messages transmitted by different processors. Third, the frequency of interprocessor communication could be reduced because each processor now executes a larger share of information. In the case of multiple copies of a program from the same task running on the same processor by context-switching, the communication overhead could be even lower because some interprocessor communication might become intraprocessor communication. Pessimistically, it is safe to assume a linear increase on execution time when a job is folded as is assumed in [9].

5.2. The RSR Algorithm

The RSR scheme consists of two components, an underlying allocation algorithm and a restriction on the number of size reductions that can be applied to a job. The RSR scheme allowing at most t times of size reduction of a job is referred to as *RSR-t* allocation. When a job is allocated using RSR, the underlying allocation algorithm is applied to locate an available submesh corresponding to the size requested by the job. If such a submesh is not located, the RSR scheme considers allocating the job to a smaller size submesh. The larger dimension of the rectangular submesh is then folded in half making the total number of processors required for the job one-half of the original size. In case a noninteger number of columns or rows results from the folding, it is rounded up to the nearest integer. The underlying allocation algorithm is then applied to allocate the job to the reduced size submesh. If such a submesh is found, the job is allocated, otherwise the folding and allocation processes repeat until the restriction on size reduction (t) is reached. If a job cannot be allocated after t times size reduction, it is queued with its original size submesh request.

The RSR scheme is very flexible in the sense that any architecture and allocation algorithm can benefit from this scheme. The reason for the restriction on job size reduction is to ensure the performance gain of reducing fragmentation is not outweighed by the loss of reduced parallelism in the application. The system administrator can determine the maximum number of foldings that a job can endure based on the system status and the workload parameters to get the optimal performance. Individual jobs can also specify their own restrictions to avoid unnecessary increase on the execution time. A job is only folded when fragmentation prevents it from execution. Jobs of all sizes have the possibility of being folded. Fragmentation is greatly reduced using RSR. The internal fragmentation and virtual fragmentation is associated with the underlying algorithm. If the chosen algorithm is free from internal and virtual fragmentation, there will be no internal and virtual fragmentation using RSR. Physical fragmentation is reduced because the fragmented nodes can be utilized with reduced-size jobs. In the extreme case, when all jobs are allowed to be executed on a single processor, there will be no fragmentation of any kind.

Complexity of the RSR allocation depends on the underlying allocation algorithm used. In a *RSR-t* allocation, the underlying allocation algorithm is called at most t times to check the availability of free processors. The worst case for the complexity of the *RSR-t* allocation algorithm would be $tC(x)$, where $C(x)$ is the complexity of the underlying algorithm. As observed later, a maximum of two to four size reductions is sufficient, thus adding very little to the complexity of the underlying allocation schemes.

6. ADAPTIVE NON-CONTIGUOUS ALLOCATION (ANCA)

Contiguous allocation schemes are prone to physical fragmentation. To alleviate this problem, noncontiguous allocation which allows jobs to be allocated on scattered nodes can be implemented. Noncontiguous allocation has the potential of improving the system performance by reducing fragmentation. We propose an *adaptive noncontiguous allocation* scheme to improve the performance of mesh-connected multicomputers.

Noncontiguous allocation schemes minimize the queuing delay of jobs by allocating the waiting jobs to noncontiguous processors. By executing jobs on noncontiguous processors, the communication latency is expected to increase. Let ΔT_{queue} and ΔT_{comm} represent the time difference between contiguous and noncontiguous allocations. The idea of developing a noncontiguous allocation algorithm is to maximize the value of $\Delta T_{\text{queue}} - \Delta T_{\text{comm}}$ so that the turnaround time of a job in Eq. (1) can benefit from a shorter queuing delay without being penalized for the communication latency. Experimental study indicates that the queuing delay can be greatly reduced resulting in a high ΔT_{queue} if the fragmentation can be reduced [8–11]. The design of faster switching devices and wormhole routing techniques have made the message passing latency insensitive to the communication distance making ΔT_{comm} negligible. Noncontiguous allocation is hence a very attractive alternative for processor allocation.

Noncontiguous allocations can be classified into two categories, totally noncontiguous and partially noncontiguous. In a totally non-contiguous allocation scheme, a job can be allocated as long as the number of available processors is sufficient for its execution. In a partially noncontiguous allocation, the nodes allocated to a job retain a certain degree of contiguity. During the execution of a process, processors assigned to a job communicate with one another. When contiguous allocation is used, the traffic is localized within the allocated submesh and causes no interprocess interference. With noncontiguous allocation, a message may have to travel through intermediate nodes assigned to other processes. Messages from different processes compete for the communication links. Messages that are blocked will have to be buffered and suffer extra overhead. Interprocess interference is expected to be less in a partially noncontiguous allocation because the local traffic in each contiguous region does not go through nodes assigned to other jobs and is less likely to collide with messages generated by other processes. Local communications within each region also cause no interference with messages generated by other processes except those messages that trespass through its territory. The study on the effect of scattered processor allocation in a wormhole routed mesh [15] agrees with this observation.

The message passing contention experiment in [7] shows that the number of communicating nodes is an important factor affecting the contention. The authors state that with small packet size and less than nine pairs of communicating nodes, the effect of message contention is virtually negligible and hence supports their noncontiguous allocation schemes. The effect of contention is more visible when the interconnection network is operated at a higher load. With diverse parallel applications, assumptions of small packet size and less intercommunicating nodes may not be valid. Another problem associated with message contention is its potential of causing network saturation. Once a network is saturated, the message latency grows infinitely large and greatly degrades the system performance.

Another issue which affects the turnaround time of a job is the geometry of the processors assigned to it. Parallel applications and algorithms are optimized to minimize the number of communication steps and the distance of communication paths. If the relative locations of the nodes for an application are violated, the communication latency is expected to increase because the communication pattern of the application is no longer optimized. Previously proposed noncontiguous allocations do not consider the geometry of the nodes and may cause extra communication overhead. This overhead can degrade the performance of the system to a great extent. Therefore, a noncontiguous allocation algorithm which carefully considers the increase of communication latency has to be derived.

Based on these observations, we derive the following conclusions regarding a good non-contiguous processor allocation scheme.

- Noncontiguous allocation should not be applied when contiguous allocation is possible to avoid the increase in T_{comm} and the possibility of saturating the interconnection network.

- When fragmentation prevents a job from being allocated, noncontiguous allocation can be applied to reduce T_{queue}.

- If a job has to be allocated noncontiguously, it is desirable to maintain some degree of contiguity to avoid an increase of T_{comm} caused by interprocess interference.

- The geometry of nodes allocated to a job has to be retained as much as possible to avoid the extra communication overhead by disrupting the optimal communication pattern.

- For a job with high communication demand or system with slow communication network, contiguous allocation prevents the introduction of communication overhead and is preferred. The system manager or the user should be provided with the flexibility of choosing the degree of noncontiguity.

The ANCA scheme is proposed to meet the above requirements of a good noncontiguous allocation scheme. It always attempts to allocate a job contiguously. When contiguous allocation is not possible, it breaks a job request into equal-sized subframes. These subframes are then allocated to available locations and thus take advantage of noncontiguous allocation. Within each subframe, the relative positions among neighboring nodes is retained.

ANCA differs from existing allocation algorithms in two aspects. First, it combines the advantages of both contiguous and noncontiguous allocation schemes. Unlike other noncontiguous allocation algorithms which allocate all jobs non-contiguously, ANCA only allocates jobs noncontiguously when physical fragmentation occurs. Second, it preserves the geometry of nodes in jobs which is important but not considered in previous noncontiguous allocations.

6.1. ANCA Scheme.

Before we present the ANCA scheme, some terminologies need to be defined for the ease of explanation. A mesh system is denoted by $M(w, h)$ where w is the number of columns and h is the number of rows of processors. A job J is denoted by $J(m, n)$ where m and n represent the number of columns and rows of processors, respectively, required for the execution of job J. A submesh can be identified by its lower-left corner which is referred to as its base.

Definition. Busy Array. For a mesh $M(w, h)$, its busy array, B, is an array in which the element $B[i, j]$ has a value 1 (0) if processor $\langle i, j\rangle$ is busy (idle).

Definition. Coverage. The coverage of an allocated submesh, I, with respect to an incoming task J is a collection of processors each of which, when used for the base of J will cause overlap between I and J. The union of the coverage of all the allocated submeshes is the coverage set of the system for the incoming task.

Coverage can be classified as left coverage and bottom coverage. Left coverage is the region of the nodes on the left side of an allocated job which cannot be served as the base of the incoming task. Bottom coverage is the region of the nodes below the left coverage and the allocated task which cannot be served as the base for the incoming job.

Definition Reject Set. The reject set with respect to an incoming task is the set of processors which can never serve as the base of an available submesh for accommodating the incoming tasks. The reject set contains the processors in the top rows and the right columns. An example of the coverage and the reject set for an allocated job is illustrated in Fig. 2 in which the processors are represented by the intersections of the horizontal and vertical lines.

Definition Coverage Array. For a mesh $M(w, h)$, its coverage array with respect to an incoming job $J = (w', h')$ is $C[w - w' + 1, h - h' + 1]$, in which $C[i, j]$ is equal to 1 (0) if processor $\langle i, j\rangle$ is (is not) in the coverage set. A 0 in the coverage array indicates the location for an available submesh for accommodating the incoming job. The coverage array has a size of $[w - w' + 1, h - h' + 1]$ which is smaller than the mesh size. This is because the processors in the reject set cannot be served as the base for the incoming task and are not included in the coverage array.

We define *adaptability* as the measure of the allocator's ability to adjust its allocation policy according to the system status. It is quantified by the number of

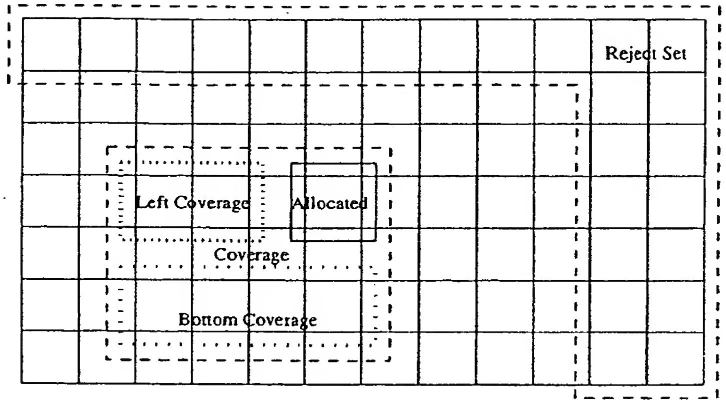

FIG. 2. The coverage and reject sets with respect to a submesh request of ⟨4, 3⟩.

times a job request is split into smaller subframes. An adaptability of 0 refers to a strictly contiguous allocation.

The ANCA scheme consists of two parts, a decomposition process and an allocation process. The decomposition process splits a job into smaller subframes for allocation when fragmentation prevents a job from being allocated to a required submesh size. The allocation process is used to check for the available processors. An ANCA scheme that allows splitting of jobs for a maximum of A times (i.e., with maximum adaptability of jobs set to A) is denoted as ANCA-A. ANCA always attempts to allocate a contiguous submesh to a job request whenever possible. If contiguous allocation fails, the decomposition process is used. A new subframe size is generated for allocation in every attempt. The allocation algorithm is then used to locate the number of subframes which can satisfy the requirements of the job. When enough subframes are found, the job is allocated. Otherwise, the decomposition and allocation process is repeated until the job is allocated or the maximum adaptability is reached. We begin our discussion of the ANCA algorithm with the decomposition and the allocation processes followed by the complete algorithm.

Decomposition process. The decomposition process splits the current subframes into smaller subframes. The generated subframe size for the next allocation attempt equals one half of the current subframe size. There are three reasons for using a subframe as the allocation unit. First, we want to maintain a certain degree of contiguity in every allocated region. By splitting a subframe into half of its current size, the sizes of all subframes are equal except those on the edges. Therefore, the subframes of a job maintain similar contiguity. Second, the geometry of nodes can be easily preserved using the subframes with a simple direct mapping mechanism. The third and most important reason for using a subframe as the allocation unit is to simplify the complexity of the allocation process. Our allocation algorithm uses a bit-array approach as used by most of the contiguous allocations. Scanning the bit-arrays for possible allocation is the most time-consuming part in the allocation algorithm. By using subframe as an allocation unit, all free subframes in the system can be located by scanning the bit-array in only one pass. In some cases, the resulting subframe size is not an integer and has to be rounded up to the nearest integer. Internal fragmentation is introduced due to the rounding up effect. This problem is solved by not allocating excessive nodes. A bookkeeping mechanism is used to keep track of the exact number of processors needed for a job and therefore avoids allocating excessive nodes to a job. Two counters are used in the bookkeeping, $x_{\text{allocated}}$ and $y_{\text{allocated}}$. A row-major order is used to allocate the subframes. Upon allocation of a subframe, a direct mapping of nodes from the original request to the subframe is performed. After a subframe is allocated, the value of $x_{\text{allocated}}$ is incremented by its size in the x dimension. If $x_{\text{allocated}}$ becomes larger than the job size in the x dimension, $x_{\text{allocated}}$ is reset to zero and $y_{\text{allocated}}$ is incremented by the subframe size in they dimension. When $y_{\text{allocated}}$ becomes larger than or equal to the job size in the y dimension, we would have the required number of subframes allocated. Fig. 3 shows how a 5 × 2 submesh is split into subframes with adaptability 1 and 2. It also demonstrates the mapping of the processors from the original request to the subframes and how excessive nodes are discarded for actual allocation.

Allocation process. The first-fit algorithm [3] is extended for ANCA. The main consideration is to allow the co-allocation of multiple subframes with minimal complexity. A coverage array is generated for the allocation of subframes. It is then scanned in a row-major order to locate the free subframes as candidate subframes for the allocation. The base of a candidate subframe is labeled as a candidate. When a candidate is found, the candidate subframe and its left coverage have to be marked as unavailable to prevent the allocation of other subframes that overlap it. The bottom coverage does not need to be marked because the scanning sequence of the rows prohibits the algorithm from locating a candidate in another candidate's bottom coverage. Thus the time required for marking the coverage

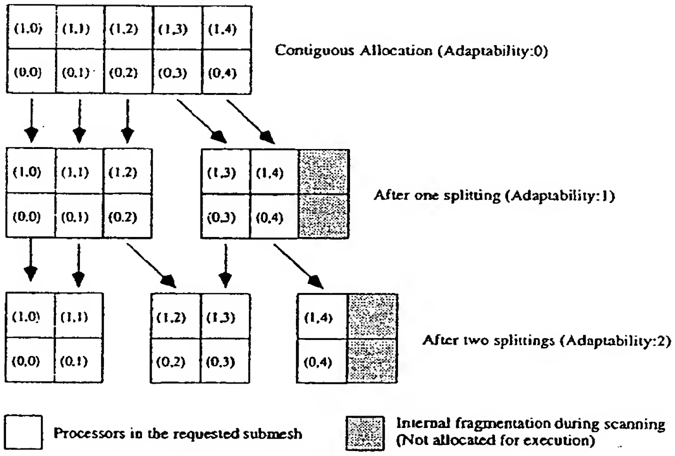
(1,0)
(1,1)
(1,2)
(1,3)
(1,4)
(0,0)
(0,1)
(0,2)
(0,3)
(0,4)
Contiguous Allocation (Adaptability:0)
After one splitting (Adaptability:1)
After two splittings (Adaptability:2)
Processors in the requested submesh
Internal fragmentation during scanning (Not allocated for execution)

FIG. 3. Example of the decomposition process.

array for bottom coverage is saved. If ANCA fails to allocate a job using a particular subframe size.it can either split the subframe and retry with a smaller subframe size or enqueue the job if the maximum adaptability is reached.

Complete algorithm. The complete algorithm for ANCA-A is presented in Fig. 4. First, the coverage array is generated for the current subframe size. After the coverage array is generated, the ANCA algorithm scans all the rows in the coverage array from bottom up. In every row, the elements are scanned from left to right. If a 1 is found at $C[i, j]$, node $\langle i, j \rangle$ is labeled as a candidate for the allocation. The candidate subframe and its left coverage is then marked as unavailable to avoid the allocation of other subframes that overlap the candidate found for allocation. The counters $x_{\mathrm{allocated}}$ and $y_{\mathrm{allocated}}$ are updated according to the bookkeeping process described earlier. The scanning for the 1's in the C array continues until $y_{\mathrm{allocated}}$ becomes larger than or equal to the required submesh size in the y dimension. If the scanning fails to find the required number of subframes for a job after exhausting the C array and the limit on the splitting is yet to be reached, the job is further decomposed and retried for allocation.

Step 1 Let (w,h) be the system size, $J=(m,n)$ be the job to be allocated. Let $(w'.h')$ be the size of the subframe for allocation attempt. Set $w'=m$, $h'=n$. Set adaptability of job, a, to 0.

Step 2 Fill the allocated submesh and their left coverage in the C array with the following procedure. Scan all the rows in $B[w,h]$. Scan each row from right to left and initialize $left.cov:=w+1$. Check $B[i,j]$. if $B[i,j]=1$, set $left.cov:=max(i-w'+1, 0)$. If $i \geq left.cov$, then set $C[i,j]=1$, else set $C[i,j]=0$.

Step 3 Fill the bottom coverage from C generated in step 2. Scan all columns in C from left to right. Scan each column from top to bottom and initialize $btm.cov:=h+1$. Check element $C[i,j]$, if $C[i,j]=1$ set $btm.cov:=max(j-h'+1.0)$. If $j \geq btm.cov$, set $C[i,j]=1$.

Step 4 Initialize the counters, $x_{allocated}$ and $y_{allocated}$ to 0.

Step 5 Scan all rows in the C array from bottom up. Scan elements in each row from left to right. At element $C[i,j]$, if $C[i,j]=1$. do nothing. Else,

(a) Set $< i, j >$ as a candidate. Mark the candidate subframe at $< i, j >$ and its left coverage as unavailable in the C array.

(b) Increment $x_{allocated}$ by w'. If $x_{allocated} \geq w'$, increment $y_{allocated}$ by h' and reset $x_{allocated}$ to 0.

(d) If $y_{allocated} \geq y'$. enough candidates are found. goto step 7. else continue the scanning from $< i, j >$.

Step 6 If $a = A$. goto step 7. Else, decompose subframe $(w'.h')$ into smaller subframes. Increase a by 1. Set (w', h') according to the size of the new subframes. If both w' or h' becomes 1 after the decomposition, goto step 8, else goto Step 2.

Step 7 Allocate the job to the candidates found in step 5. To allocate the next job in the queue, goto step 1.

Step 8 Allocation failed. Put the job in the system queue and wait for the departure of an executing job for retrial.

FIG. 4. The ANCA algorithm.

The processor relinquishment is simple. When a job departs, the busy array is updated according to the nodes released by the departing process. The system is then signaled for allocation of jobs waiting in the queue.

6.2. *Complexity Analysis of the ANCA Algorithm*

The space complexity of the ANCA algorithm involves storage of the busy array, the coverage array, and the temporary storage for the candidate subframes. It is the same as the first-fit algorithm except that in ANCA, additional storage is required for the candidates. The time complexity for ANCA in one iteration is slightly worse than the first-fit algorithm because of the extra time taken for the marking of the candidate subframes. The marking of the candidates in one iteration takes $O(mn)$. For allocation of an $m \times n$ job in a $w \times h$ mesh, the big O representation for the time complexity is $O(wh + mn) = O(wh)$. Therefore the time complexity of the ANCA-A algorithm to allocate a job is equal to $O(Awh + Amn)$.

7. PERFORMANCE ANALYSIS

7.1. *Simulation Environment*

Two workload models have been used in [8, 19] to characterize the jobs in a parallel computer system, the correlated workload and the uncorrelated workload. The correlated workload model assumes that the work demand of a job does not vary with the number of processors allocated to the job. Therefore, jobs executed on more processors are likely to have shorter execution times. This workload model agrees with the assumption used in Amdahl's law [17] for speedup. The uncorrelated workload model assumes that the work demand of a job scales with the number of processors used for its execution. This assumption is justified by Gustafson's law [20] which states that many scientific computations can be scaled up to obtain higher accuracy when more processors are used. RSR allocation reduces the number of processors allocated to a job andincreases the execution time of the job. It is therefore safe for us to adopt the correlated workload in the simulation of the RSR scheme because the increase of execution time in the correlated workload is more significant than that with the uncorrelated workload assumption. The RSR scheme should perform better or equally well when the uncorrelated workload is assumed.

The performance of noncontiguous allocation schemes depend heavily on the communication latency of a job. Because of the rapidly advancing technology and diversity of applications, the cost of communication latency is hard to estimate. It may be in appropriate and misleading to evaluate a noncontiguous allocation scheme with an approximated communication latency. Therefore, we evaluate the ANCA scheme for two extreme cases. An optimistic scenario illustrates the best performance that ANCA can provide. A pessimistic (if not the worst) case is simulated to consider thelimit on the potential gain of ANCA. The actual performance of the ANCA scheme can then be predicted from the results of these two scenarios. The system simulated is a 32×32 mesh. Jobs are assumed to arrive in a Poisson process with the service time assumed to be exponentially distributed. To demonstrate the performance of different allocation schemes in a dynamic environment. different arrival rates are simulated. The arrival rate is calculated from the *traffic ratio* which is defined as the ratio of arrival rate to service rate. We choose a service rate of 0.2 jobs per unit time for our simulation. The size of a job in each dimension follows the same distribution independently. Two different distributions for the submesh sizes are simulated. The uniform distribution assumes the side lengths of a job range from 1 to 32 with equal probabilities. We also assumed a truncated normal distribution for the side lengths of a job in which the mean is set to 16.5 with a variance of 6.6. Each run collects the data for the first 10,000 job completions.

The metric of interest in these simulations are fragmentation and the average turnaround time of a job. To include all kinds of fragmentation, we define fragmentation as the summation of the ratio of available processors to the total number of processors at each allocation failure divided by the total number of allocation attempt. Therefore, fragmentation is a function of the system load and efficiency of the allocation scheme. At the same traffic load, a better allocation scheme should produce less fragmentation. The average turnaround time is the time between submission of a job and its completion. This is a direct measure of the system performance. Using the two proposed schemes, execution time of individual jobs may be increased. In RSR, this is caused by size-reduction while in ANCA this is due to increased communication latency. However, the queuing delays in both cases are reduced. The average turnaround time of jobs is therefore more important to calibrate the performance of these two schemes.

7.2. *Impact of the RSR Scheme*

Figure 5 illustrates the fragmentation of RSR schemes with different restrictions on size reductions. Results are shown for uniform and normal job size distributions. Both the distributions show similar trends in the results. As the maximum size reduction *t* increases, the fragmentation is reduced in two ways. First, the physical fragmentation is reduced because scattered nodes can be utilized to execute size-reduced jobs. Second, the insufficient resource fragmentation is also alleviated because jobs are able to run with a smaller number of processors. In the extreme case of *RSR*-10, any job can be executed on a single node. There is no fragmentation of any kind in this case.

Because of the reduced fragmentation, RSR has a shorter average turnaround time for jobs as shown in Fig. 6. Using RSR, a job is executed as early as the size reduction restriction allows. Therefore, it is less likely to block other jobs. The improvement is more visible under high system load where the blocking effect is more pronounced and causes the average turnaround time to increase substantially. With RSR, this effect is reduced and therefore the average turnaround time is also reduced.

Another observation made from these results is the tradeoff between the larger operational range and the lower average turnaround time. When the system load is high, a system needs to accommodate as many jobs as possible in order to avoid saturation. Allowing more size reduction makes this possible. However, as noticed

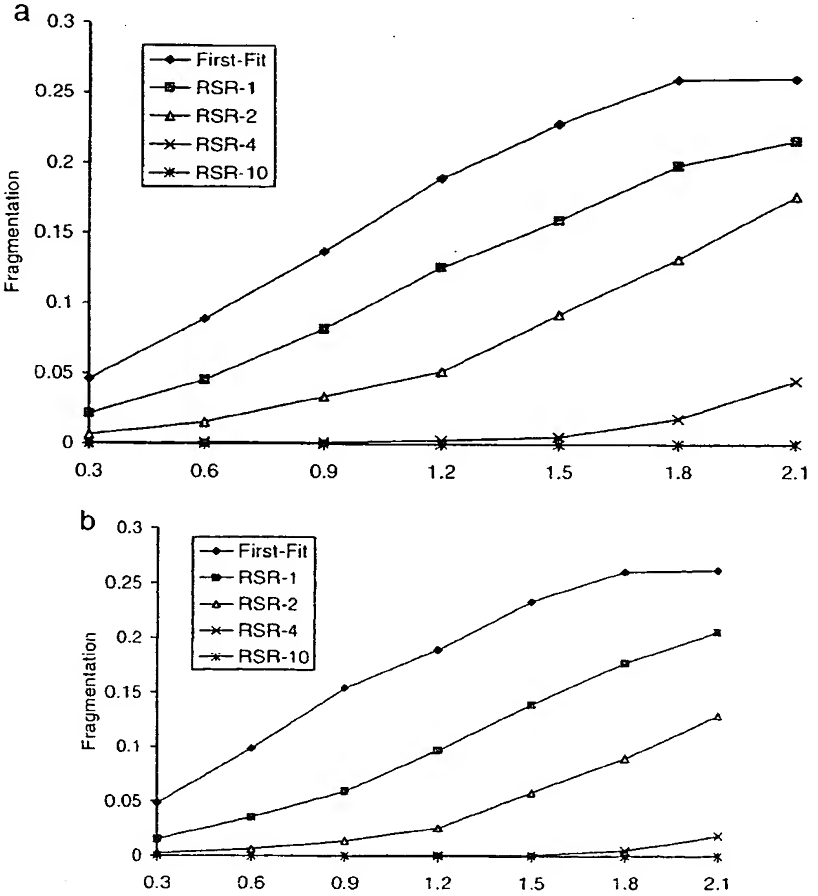

FIG. 5. Fragmentation vs system load for RSR schemes in a 32×32 mesh. (a) Uniform job-size distribution; (b) Normal job-size distribution.

in the RSR scheme, allowing a smaller number of size reductions provides shorter turnaround time under low to medium load. This is because the allocations allowing more size reductions tend to reduce the size of a job more often. As fragmentation is not serious under these loads, execution time is the dominant factor in the average turnaround time. Therefore, allowing fewer size reductions avoids the unnecessary job size reduction and provides a better performance under such loads. On the other hand, the system performance improves rapidly with a small number of size reductions. The performance improvement of allowing more size reductions is not significant at low to medium load. Greater number of size reductions, although allowing the mesh to be operated at a higher system load, incurs a very high turnaround time at high traffic ratio because many jobs encounter severe size reduction. The execution time for a job requiring 1024 nodes may be increased 1024 times if it is folded down to a single node. Moreover, the size reduction may be

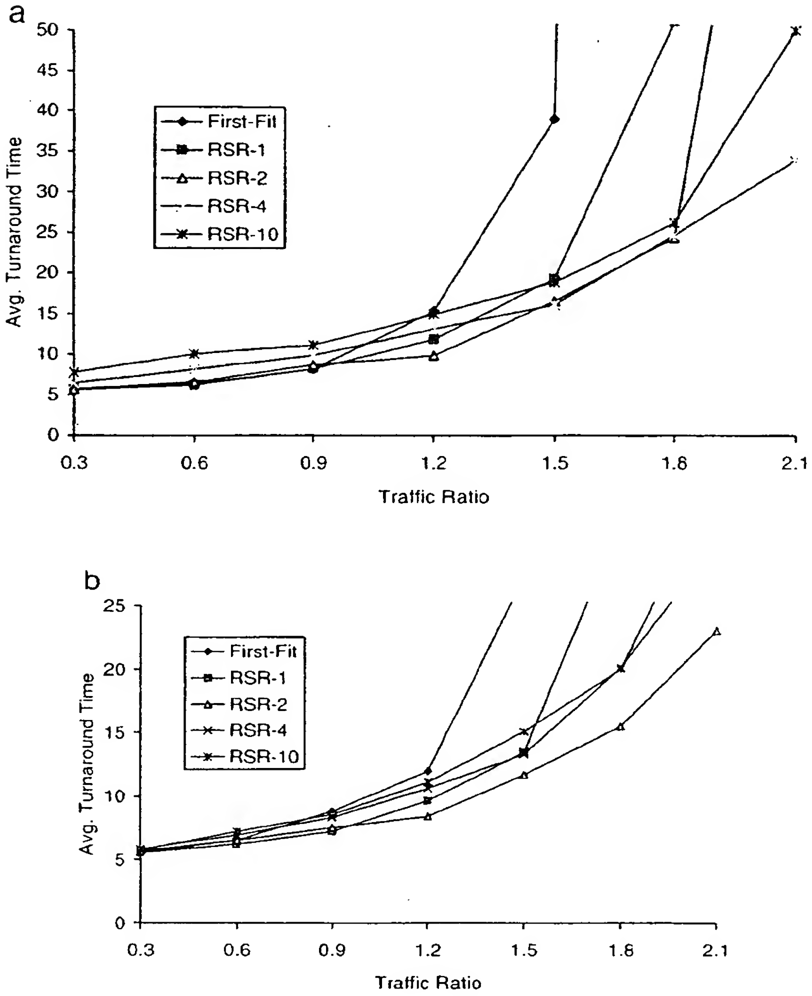

FIG. 6. Average turnaround time vs system lad for RSR schemes in a 32 × 32 mesh. (a) Uniform job-size distribution; (b) Normal job-size distribution.

restricted by the memory space available at the nodes. In our experiments, we have assumed sufficient memory space at the nodes to demonstrate the effect of various degrees of size reductions. It is preferable to allow a small number of size reductions, such as two or four, to improve the system performance while avoiding an unnecessary overhead caused by a large number of size reductions.

7.3. Performance of ANCA Scheme

The ANCA scheme is evaluated under two scenarios. An optimistic scenario assumes that no communication overhead is caused by noncontiguous allocation. In this scenario, the execution time of a job remains unchanged regardless of whether it is allocated noncontiguously or not. This scenario reflects the performance of systems with an efficient interconnection network or jobs with low communication demand. Communication overhead is exaggerated in the pessimistic scenario to estimate the performance limit of ANCA policy in an operational mesh. A previous study [13] indicates an important characteristic for the message delay of wormhole routed meshes. In a wormhole routed mesh, the average message delay is relatively constant with a low arrival rate of messages. It grows slowly when the arrival rate of message increases. A higher message arrival rate means more contention for the communication links in the network. Once the message arrival rate increases beyond the system's capability, the network saturates and the message delay goes out of bounds. Typically the effect of message contention increases the average packet delay by less than three folds [13]. We devise our pessimistic scenario based on this characteristic. Another factor involved in communication overhead is the ratio of communication time over the total execution time for a job. The ratio of communication time to computation time for a job in multicomputers is usually low. Our pessimistic scenario is devised in an attempt to capture the behavior of a near-saturation mesh. Therefore, the communication time of a job is set at a high value of 30% of its total execution time. A noncontiguously allocated job is penalized by increasing its communication latency by three times. By exaggerating the message transfer delay and the communication demand of jobs, the pessimistic scenario reflects the behavior of a near-saturation mesh filled with communication-intensive jobs. The actual performance of a particular ANCA algorithm can be predicted with the two curves drawn for its turnaround time in the optimistic and pessimistic scenarios. However, this prediction is only valid when the communication network is not saturated. Therefore, we also discuss the percentage of jobs being allocated contiguously which is directly related to the possibility of network saturation.

7.3.1. *Fragmentation of ANCA schemes.* Fragmentation of various ANCA schemes in the optimistic scenario is plotted in Fig. 7. Higher adaptabilities have less fragmentation. Adaptability zero is equivalent to the first-fit algorithm. Previously proposed noncontiguous allocation algorithms such as MBS and Naive allocations [7] have identical fragmentation curves as ANCA with adaptability 10. Our results differ from the one shown in [7] in which MBS and Naive are said to be free of any type of fragmentation. This is because of the different definitions used for fragmentation. The results shown in [7] do not include the insufficient resource fragmentation.

Fragmentation provides a measure of the system performance in most cases but it could also be misleading. Lower fragmentation means higher utilization. However, since partial contiguity and geometry of nodes are important to lower the communication overheads, high utilization achieved by noncontiguous allocation does not necessarily lead to better performance. Processors may be occupied by jobs waiting for delayed messages and do not actually contribute to real execution of any job. With the same fragmentation, ANCA should provide better performance than other noncontiguous schemes because of the partial contiguity and preservation of geometry of the allocated nodes.

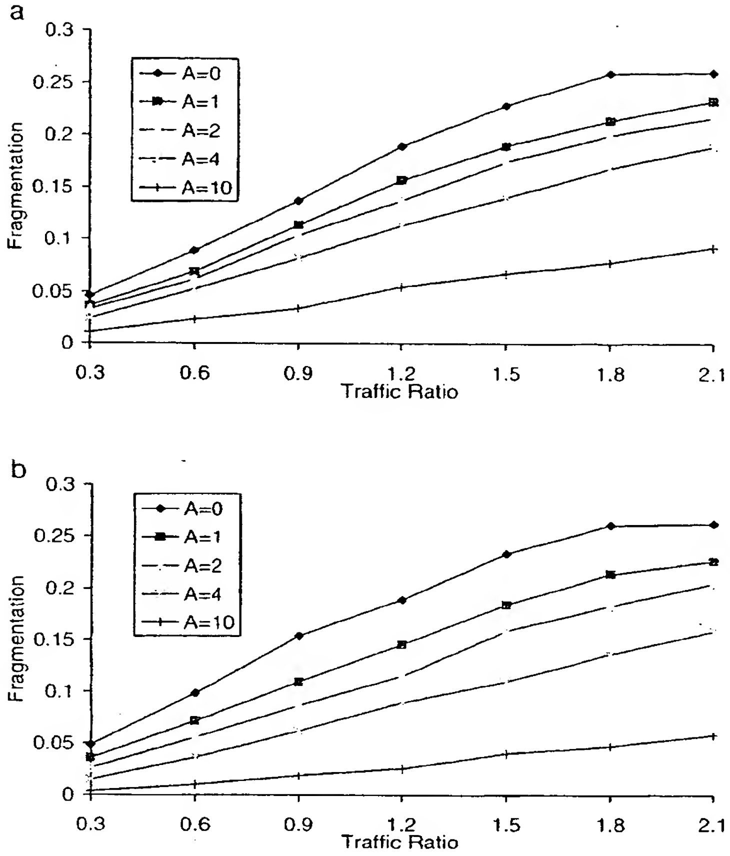

FIG. 7. Fragmentation of various ANCA schemes. (a) Uniform job-size distribution; (b) Normal job-size distribution.

The fragmentation of ANCA schemes in both scenarios are compared in Fig. 8 at traffic ratio of 2.1. The values obtained from the pessimistic scenario are higher than the values for the same adaptability obtained in the optimistic scenario. This is because job execution tends to be longer in the pessimistic scenario due to the longer communication latency. Therefore, allocation failure occurs more often and causes higher fragmentation.

7.3.2. *Predicting the Average Turnaround Time of ANCA.* Figure 9 plots the average turnaround time for various ANCA schemes in the optimistic scenario. Both uniform and normal distributions of job sizes show a similar trend. As expected, higher adaptability provides lower turnaround time because of the reduction of fragmentation. The difference is noticeable with a small increase of adaptability from 0 to 1 or 2. The improvement of average turnaround time is especially significant at traffic ratio higher than 0.9. For lower traffic, queuing is seldom encountered and the turnaround time is mainly dependent on the execution time of the

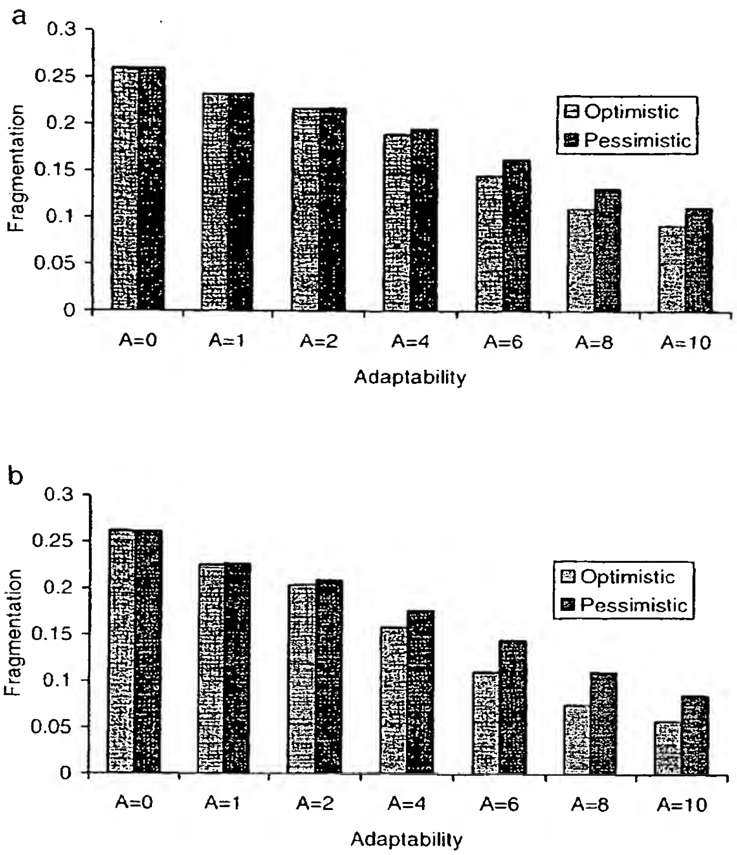

FIG. 8. Fragmentation of ANCA schemes in both scenarios. (a) Uniform job-size distribution; (b) Normal job-size distribution.

jobs. At higher traffic, the queuing delay becomes a dominating factor in the average turnaround time and hence ANCA performs better.

Figure 10 shows the average turnaround time for different adaptability when the pessimistic scenario is assumed. Although the fragmentation results shown in Fig. 8 do not show a significant difference between the optimistic and pessimistic scenarios, the pessimistic scenario fails to improve the performance of mesh system efficiently. This is because of the large communication overhead assumed for the pessimistic case. Noncontiguously allocated jobs stay in the system for a longer period of time and cause succeeding jobs to wait.

Because of the rapidly advancing technology and diverse applications, it is inappropriate to predict the performance of a noncontiguous allocation scheme based on any one scenario. Therefore, the performance of the ANCA scheme in both scenarios has to be considered when predicting its actual performance. The average turnaround time of a system implementing the ANCA algorithm would lie in

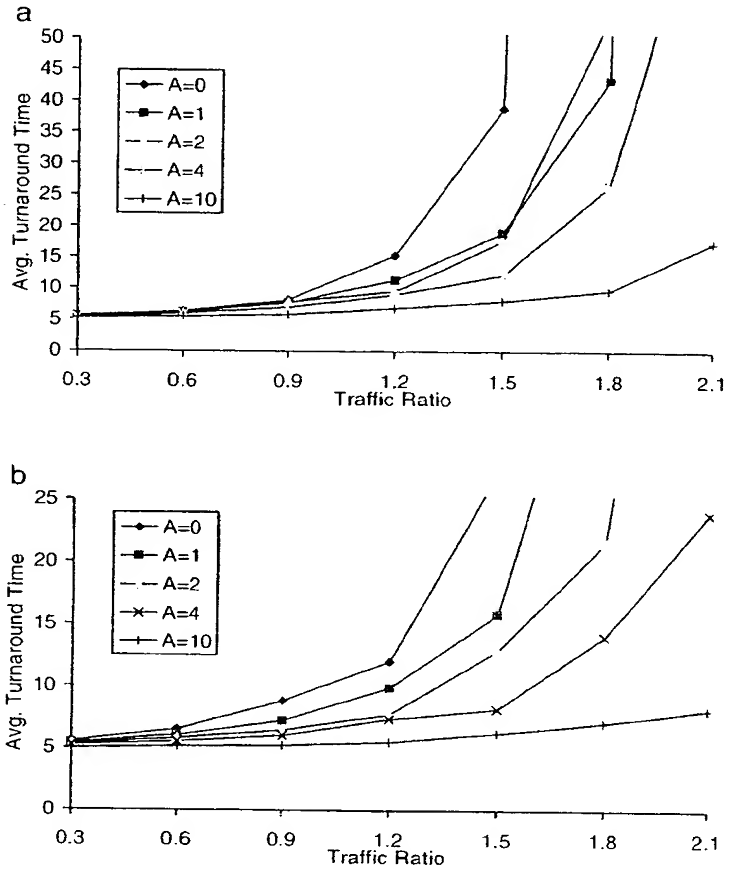

FIG. 9. Average turnaround time for ANCA with different adaptability in the optimistic scenario. (a) Uniform job-size distribution; (b) Normal job-size distribution.

between the two curves: the optimistic case and the pessimistic case, provided the network is not saturated.

Figure 11 illustrates a comparison of the expected performances of ANCA-1 and ANCA-4 compared to the first-fit algorithmfor uniform job size distribution. The pessimistic scenario results of ANCA-1 are close to the first-fit allocation. In some cases, first-fit has a shorter average turnaround time than the pessimistic case. Both the optimistic and pessimistic scenarios of ANCA-4 have a shorter turnaround time than the first-fit algorithm. Similar observations can be made in Fig. 12 for the normally distributed jobs. ANCA-4 is guaranteed to improve the system performance over the first-fit algorithm while this may not be true for ANCA-1.

7.3.3. *Possibility of saturating the network by ANCA.* The performance of a noncontiguous allocation algorithm may not be better than a contiguous allocation considering the higher possibility of saturating the interconnection network. When

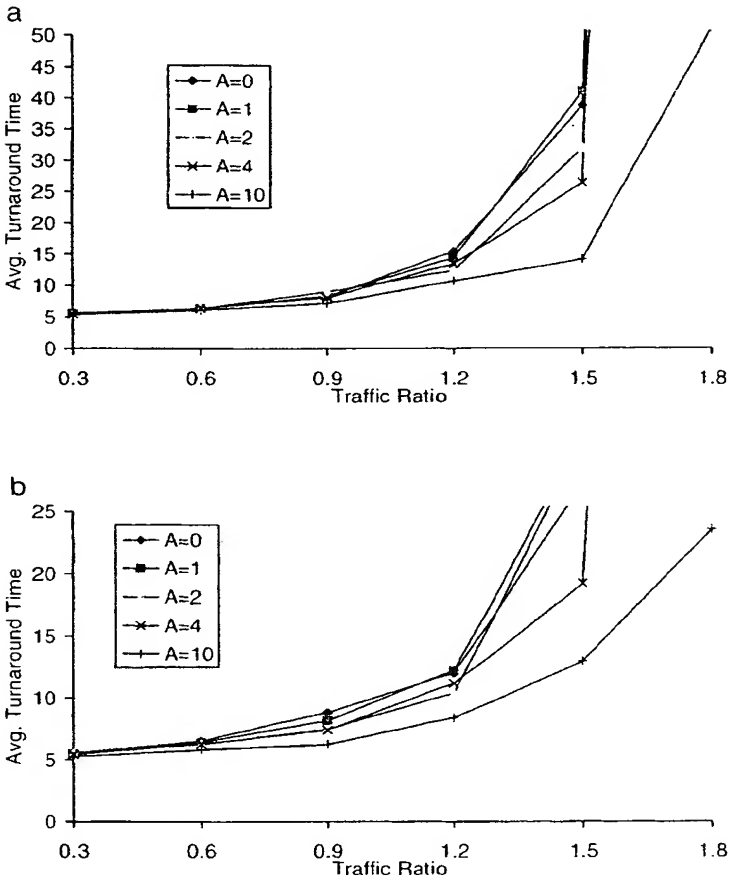

FIG. 10. Average turnaround time for ANCA with different adaptability in the pessimistic scenario. (a) Uniform job-size distribution; (b) Normal job-size distribution.

an interconnection network is saturated, the message passing latency grows infinitely and results in a high degradation of the system performance. We compare the percentage of jobs being allocated contiguously under the pessimistic assumption in Fig. 13. The results of the optimistic scenario show a similar trend and are omitted. The first-fit algorithm is a strictly contiguous allocation scheme and hence allocates all jobs contiguously. Greater adaptability allows jobs to be allocated in smaller subframes and therefore allocates fewer jobs contiguously. The percentage of contiguous jobs drops when the traffic increases because fragmentation occurs more often and requires more jobs to be split. In the worst case, the ANCA algorithm allocates 34.6% of jobs contiguously for the uniform jobs and 11.4% for the normally distributed jobs. The higher percentage of contiguous jobs reflects a lower communication overhead and can be interpreted as a lower possibility of saturating the interconnection network. With a lower adaptability set for the system, we can

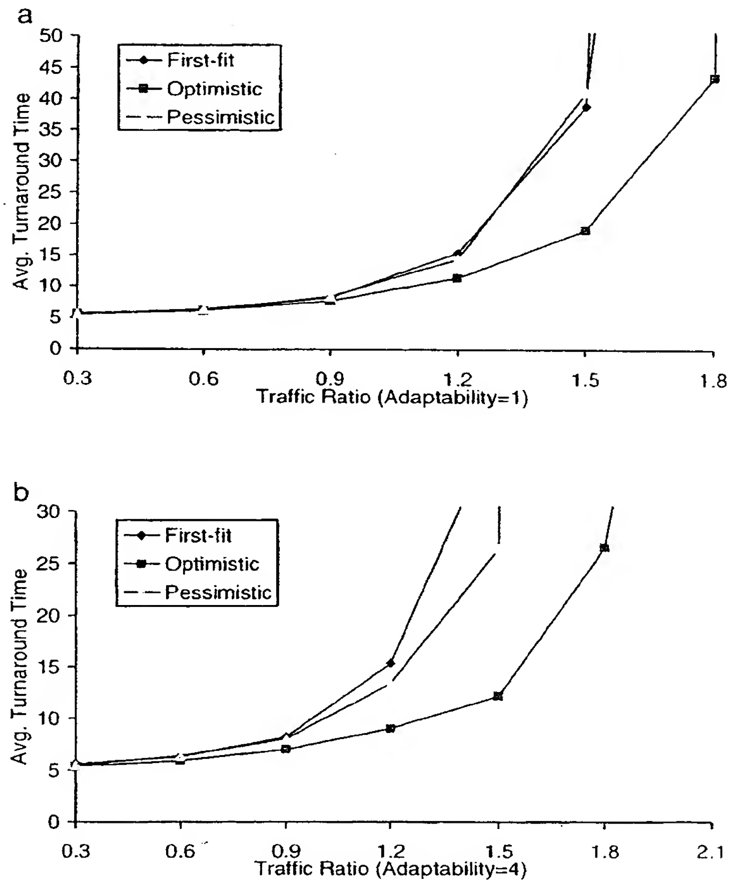

FIG. 11. Predicting the average turnaround time for ANCA. (a) ANCA-1; (b) ANCA-4.

ensure a high percentage of contiguous jobs and therefore prevent the interconnection network from saturating. Previously proposed noncontiguous allocation schemes do not have this ability to maintain a high percentage of contiguously allocated jobs and therefore have the potential to saturate the interconnection network. For example, the MBS scheme provides partial contiguity and is believed to have the best performance among the algorithms proposed in [7]. In MBS, a job can be allocated without changing its submesh request only when its size equals a buddy and when a buddy of that size is available. In a 32 × 32 mesh, only square jobs with side lengths equal to 1, 2, 4, 8, 16, and 32 can be allocated contiguously. Assuming a uniform distribution, the probability of having such a submesh request is equal to $6/1024 \cong 0.6\%$. This probability has to be multiplied with the probability of having an available buddy of the equal size and therefore the actual percentage of jobs that can be allocated contiguously in the MBS scheme is much lower than 0.6%. It is quite possible for the network to saturate with many jobs allocated noncontiguously, and in addition, have their geometry altered.

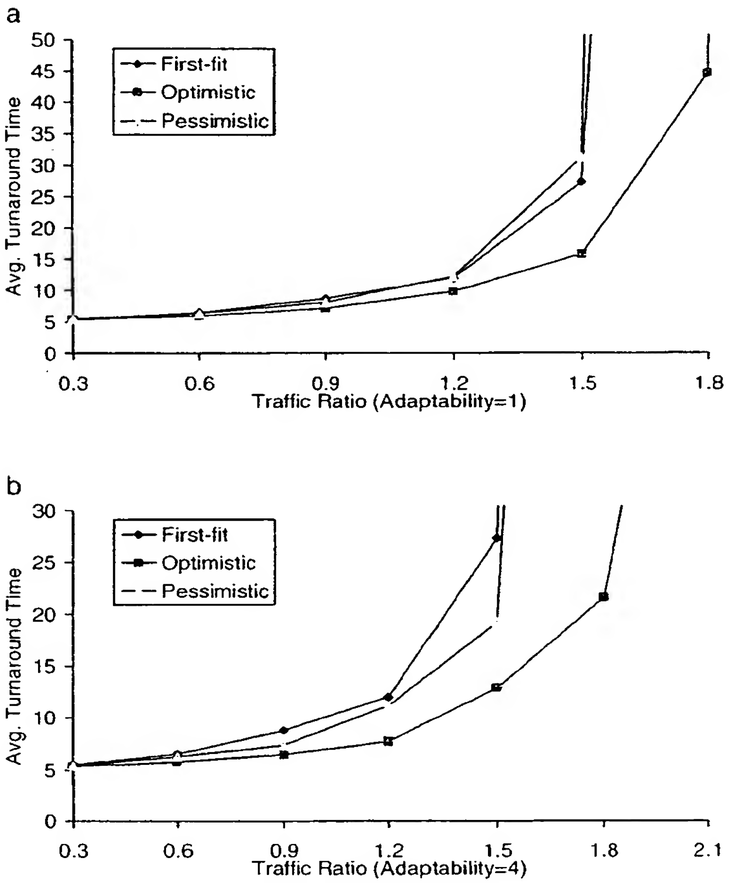

FIG. 12. Predicting the average turnaround time for ANCA. (a) ANCA-1; (b) ANCA-4.

7.4. Further Discussion

Both schemes discussed in this paper significantly reduce the fragmentation. As a result, average turnaround times for jobs are also reduced. However, it is difficult to compare these two schemes directly. In most cases, the average turnaround time collected for RSR schemes lies in-between the two curves drawn for the ANCA schemes. It is not clear which scheme can guarantee a better performance.

Each scheme has certain disadvantages. The RSR scheme reduces the size of a job which may not be possible for some applications. Memory threshold is another problem associated with the RSR scheme. When a job is allocated to a smaller submesh, the storage requirement on every assigned node is increased. It is possible to exhaust the available memory by size-reduction and result in the heavy swapping of memory. Performance of the ANCA scheme depends on the speed of the interconnection network and the communication characteristics of jobs. If the

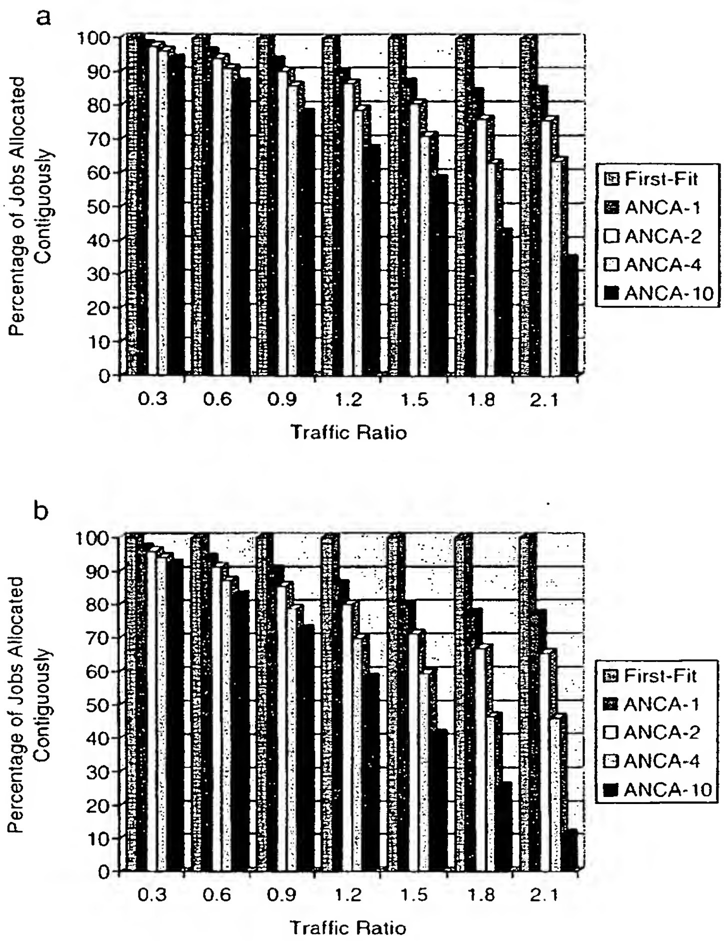

FIG. 13. Percentage of contiguously allocated jobs. (a) Uniform job-size distribution; (b) Normal job-size distribution.

interconnection network is slow or jobs are communication-intensive, the ANCA scheme may fail to deliver better performance.

Judicious selection of maximum size reduction for RSR or adaptability for ANCA is important. Our results suggest using small values for both parameters because the performance gains may be offset by the tradeoffs with a large number of size reductions or high adaptabilities. In RSR, the tradeoff is the increased execution time, and in ANCA it is the communication overhead. By using small values for these parameters, the above problems of memory space and communication overheads can also be avoided. Both RSR and ANCA can be implemented to allow individual jobs flexibility in specifying the number of size reductions allowed or adaptabilities. With this flexibility, a job requiring a large memory space can request the allocator to not fold it and communication-intensive jobs can request to be allocated with a higher degree of contiguity. It is also possible to combine the two schemes to support a system with diverse applications.

8. CONCLUDING REMARKS

Fragmentation problems can limit the performance of mesh-connected multicomputer systems. Based on the causes, fragmentation is classified into internal, insufficient resource, virtual, and physical fragmentation. Conventional allocation schemes fail to reduce the effect of physical fragmentation and hence provide limited performance. Two novel processor allocation schemes which effectively address the problem of physical fragmentation are evaluated in this paper. The restricted size reduction (*RSR*) method reduces the size of a job when physical fragmentation prevents the job from being executed. As a side effect, it also reduces the insufficient resource fragmentation. The adaptive noncontiguous allocation (*ANCA*) scheme allows jobs to be allocated noncontiguously so that physically fragmented processors can be utilized.

There are performance tradeoffs associated with both schemes. For the RSR scheme, folded jobs have a longer execution time and may cause heavy swapping. For the ANCA scheme, systems with slow interconnection networks or communication intensive jobs may suffer. Therefore, it is important to choose proper system parameters. The size reduction or adaptability can be specified at the system level or by individual jobs. This flexibility allows the system administrator to choose an optimal value while allowing jobs requiring large memory space or communication intensive jobs to avoid unnecessary overhead. The flexibility along with the performance and low complexity make these two schemes attractive for real implementation.

REFERENCES

1. K. Li and K. H. Cheng, A two-dimensional buddy system for dynamic resource alocation in a partitionable mesh connected system, *J. Parallel Distrib. Comput.* 12 (1991), 79–83.
2. P. J. Chuang and N. F. Tzeng, Allocating precise submesh in mesh-connected systems, *IEEE Trans. Parallel Distrib. Systems* (Feb. 1994), 211–217.
3. Y. H. Zhu, Efficient processor allocation strategies for mesh-connected parallel computers, *J. Parallel Distrib. Comput.* 16 (1992), 328–337.
4. J. Ding and L. N. Bhuyan, An adaptive submesh allocation strategy for two-dimensional mesh connected systems, *Proc. Int. Conf. Parallel Process.* II (Aug. 1993), 193–200.
5. D. D. Sharma and D. K. Pradhan, A fast and efficient strategy for submesh allocation in mesh-connected parallel computers, *in* "Proc of the 5th IEEE Symp. on Parallel and Distributed Processing," (Dec. 1993), 682 693.
6. T. Liu, W.-K. Huang, F. Lombardi, and L. N. Bhuyan, A submesh allocation scheme for mesh-connected multiprocessor systems, *Proc. Int. Conf. Parallel Process.* II (Aug. 1995), 159–163.
7. W. Liu, V. Lo, K. Windish, and B. Nitzberg, Non-continuous processor allocation algorithms for mesh-connected multicomputers, *IEEE Trans. Parallel Distrib. Systems* (July 1997), 712–726.
8. P. Krueger, T. Lai, and V. A. Dixit-Radiya, Job scheduling is more important than professor allocation for hypercube computers, *IEEE Trans. Parallel Distrib. Systems* 5 (May 1994), 488–197.

9. C. Hu and C. R. Das, Limit allocation: An efficient processor management scheme for hypercubes, *Int. Conf. on Parallel Process.* II (1994), 143 150.
10. C. Y. Chang and P. Mohapatra, An adaptive job allocation method for the directly-connected multicomputer systems, *in* "International Conference on Distributed Computing Systems," (May 1996).
11. P. Mohapatra, C. Yu, and C. R. Das, A lazy scheduling scheme for hypercube computers, *J. Parallel Distrib. Comput.* 27 (1995), 26–37.
12. B. S. Yoo, C. R. Das, and C. Yu, Processor management techniques for mesh-connected multiprocessors, *Proc. Int. Conf. Parallel Process.* II (Aug. 1995), 105–112.
13. J. H. Upadhyay, V. Varavithya, and P. Mohapatra, Efficient and balanced adaptive routing in two-dimensional meshes, *in* "Proc. of the IEEE Symp. on High-Performance Computer Architecture," (Jan. 1995), pp. 112–121.
14. S. Q. Moore and L. M. Ni, The effect of network contention on processor allocation strategies, *in* "Proc. of the 1996 International Parallel Processing Symposium," (April 1996).
15. D. Min and M. W. Mutka, Effect of job interactions due to scattered processor allocations in 2-D wormhole networks, *in* "Proc. of Int. Conf. on Parallel and Distributed Computing Systems," (Sept. 1995), pp. 262–267.
16. "nCube2 Programmer's Guide," nCUBE, 1992.
17. G. Amdahl, Validity of the single-processor approach to achieving large scale computing capabilities, *in* "Proc. AFIPS Conf.," (1967), pp. 483–485.
18. X. H. Sun and L. M. Ni, Scalable problems and memory-bound speedup, *J. Parallel Distrib. Comput.* 19 (1993), 27–37.
19. S. T. Leutenegger and M. K. Vernon, The performance of multiprogrammed multiprocessor scheduling policies, *in* "Proc. ACM SIGMETRICS Conf.," (1990), pp. 226–236.
20. J. L. Gustafson, Reevaluating Amdahl's law, *Comm. Assoc. Comput. Mach.* (May 1988), 532–533.

CHUNG-YEN CHANG received the B.S. degree from National Sun Yat-Sen University, Taiwan, R.O.C., in 1990 and the M.S. degree from University of Nebraska-Lincoln in 1992, both in electrical engineering. He graduated with a Ph.D. degree in computer engineering from Iowa State University in 1997, receiving a graduate research excellence award from the university. Since then, he has been with the System Performance Department at Amdahl Corporation as a computer performance analyst. His research interests include computer architecture, parallel and distributed systems, and performance evaluation.

PRASANT MOHAPATRA received the B.S. degree in electrical engineering from Regional Engineering College, Rourkela, India, in 1987 and the Ph.D. degree in computer engineering from the Pennsylvania State University in 1993. Currently, he is an associate professor in the Department of Electrical and Computer Engineering at Iowa State University. During the summer of 1998, he worked as a visiting scientist for Panasonic Technologies at Princeton, NJ. His research interests include multimedia systems, storage architecture, computer architecture, parallel and distributed systems, performance evaluation, and fault-tolerant computing.